Computation-Communication Overlap Techniques for Parallel Spectral Calculations in Gyrokinetic Vlasov Simulations

Shinya MAEYAMA, Tomohiko WATANABE[1)], Yasuhiro IDOMURA, Motoki NAKATA, Masanori NUNAMI[1)] and Akihiro ISHIZAWA[1)]

Japan Atomic Energy Agency, Rokkasho 039-3212, Japan
[1)]*National Institute for Fusion Science, Toki 509-5292, Japan*

(Received 11 July 2013 / Accepted 28 August 2013)

One of the important phenomena in magnetically-confined fusion plasma is plasma turbulence, which causes particle and heat transport and degrades plasma confinement. To address multi-scale turbulence including temporal and spatial scales of electrons and ions, we extend our gyrokinetic Vlasov simulation code GKV to run efficiently on peta-scale supercomputers. A key numerical technique is the parallel Fast Fourier Transform (FFT) required for parallel spectral calculations, where masking of the cost of inter-node transpose communications is essential to improve strong scaling. To mask communication costs, computation-communication overlap techniques are applied for FFTs and transpose with the help of the hybrid parallelization of message passing interface and open multi-processing. Integrated overlaps including whole spectral calculation procedures show better scaling than simple overlaps of FFTs and transpose. The masking of communication costs significantly improves strong scaling of the GKV code, and makes substantial speed-up toward multi-scale turbulence simulations.

Keywords: computation-communication overlap, parallel fast Fourier transform, parallel spectral calculation, MPI/OpenMP hybrid parallelization, Vlasov simulation

DOI: 10.1585/pfr.8.1403150

1. Introduction

Turbulent transport is one of the most important issues in magnetically-confined plasma researches. Pressure gradients of confined plasma destabilize micro-instabilities, and then the destabilized micro-fluctuations drive plasma turbulence via nonlinear coupling between perturbations of electric potentials and plasma distribution functions. The plasma turbulence causes particle and heat transport perpendicular to magnetic fields and degrades plasma confinement. Typical wavelengths of the plasma turbulence are of the order of gyroradii of charged particles, while its time scales are slower than gyrations.

To treat plasma turbulence, the gyrokinetic theory has been developed. The equations describe time evolution of gyrophase-averaged distribution functions and electric potentials with retaining finite gyroradius effects. A lot of numerical simulations based on the gyrokinetic theory have been carried out and have contributed to understandings of plasma turbulence [1]. The development of numerical techniques for gyrokinetic simulations has continued to improve their applicability, accuracy and efficiency. Gyrokinetic simulations require expensive computational resources, since they have to solve time evolution of gyrophase-averaged distribution functions in five-dimensional (5D) phase space. We have investigated ion-scale plasma turbulence by using our gyrokinetic simulation code GKV [2] on tera-scale supercomputers. One of important intrinsic features of plasma turbulence is its multi-scale physics, which includes temporal and spatial scales of electrons and ions. Each of two perpendicular directions requires the square root of the ion-to-electron mas ratio (~ 43) times finer resolution (therefore, the total perpendicular resolution is ~ 1836 times finer) than that of ion-scale turbulence. To deal with this numerically challenging problem, peta-scale computing is necessary.

To realize efficient computations of multi-scale turbulence simulations, parallelization of the high-resolution perpendicular space is required. In the GKV code, perpendicular dynamics is solved by using the spectral method [3] with Fast Fourier Transform (FFT) algorithms [4]. The most commonly used parallel multi-dimensional FFT is the transpose-split method [5], and one can find many literatures about the spectral or pseudo-spectral methods with parallel FFTs (e.g., 2D FFTs [6], 3D FFTs with the slab decomposition [7] and with the pencil decomposition [8]). While the transpose communications may degrade scalability, it is reported that the overlap of communications and computations improves efficiencies [5, 9, 10].

To make the GKV code run efficiently on peta-scale supercomputers, we extended the GKV code in two steps. First, 3D domain decomposition is extended to 4D one by adding the perpendicular-space decomposition, which enables us to employ a larger number of cores. The transpose communications required for parallel 2D FFTs are implemented by using a simple blocking collective communi-

author's e-mail: maeyama.shinya@jaea.go.jp

cation, which makes implementation easy and the use of specific algorithms possible, like collective communications optimized for the K computer [11]. Second, transpose communications and FFT computations are overlapped by employing a communication thread in a hybrid parallel model of Message Passing Interface (MPI) and Open Multi-Processing (OpenMP), which enables overlaps of computations and blocking communications as well as non-blocking communications [12]. Masking of communication costs significantly improves strong scaling of the perpendicular-space parallelization.

The paper is organized as follows. Section 2 explains governing equations and simulation models employed in the GKV code. Section 3 describes parallelization methods developed for peta-scale computing. Section 4 represents performance analysis of the wave-number-space decomposition, and the overlap methods. Section 5 demonstrates speed-up of the GKV code on the K computer. Finally, results are summarized in Sec. 6.

2. The GKV Code

The GKV code is originally developed to investigate ion-temperature-gradient-driven turbulence with the adiabatic electron approximation. Extensions of the code for treating both of kinetic ions and electrons have recently been done [13]. Since the employed numerical algorithms are principally the same, we treat gyrokinetic equations with the adiabatic electron approximation in the following manuscript.

2.1 Governing equations

The GKV code solves the so-called δf gyrokinetic equations, where the distribution function is split into the equilibrium part $F_{\rm M}$ and the perturbed part δf. Then the time evolution of the gyrophase-averaged perturbed ion distribution function $\delta\bar{f}_{\rm i}(\boldsymbol{r}, v_\parallel, \mu; t)$ is described by the gyrokinetic Vlasov equation,

$$\left[\frac{\partial}{\partial t}+\left(v_\parallel\frac{\boldsymbol{B}}{B}+\boldsymbol{v}_{\rm d}+\boldsymbol{v}_{\rm E}\right)\cdot\boldsymbol{\nabla}-\frac{\mu\nabla_\parallel B}{m_{\rm i}}\frac{\partial}{\partial v_\parallel}\right]\delta\bar{f}_{\rm i}=\mathcal{S}+C, \tag{1}$$

where $v_\parallel$, $\boldsymbol{v}_{\rm d}$ and $\boldsymbol{v}_{\rm E}$ are the velocity parallel to the confinement magnetic field, the perpendicular magnetic drift velocity and the perpendicular $\boldsymbol{E}\times\boldsymbol{B}$ drift velocity due to electric potential perturbations. The term with the magnetic moment μ and the ion mass $m_{\rm i}$ represents parallel acceleration by the mirror force. The linear term associated with the equilibrium distribution $\mathcal{S}$ contains contributions of the parallel electric field and equilibrium pressure gradients, which drive micro-instabilities and plasma turbulence. The model collision operator C is friction and diffusion operators in velocity space $(v_\parallel, \mu)$. The perturbed electric potential ϕ is given by the gyrokinetic quasi-neutrality equation with the elementary electric charge e and the ion equilibrium temperature $T_{\rm i}$,

$$\int\left[\delta\bar{f}_{\rm i}-\frac{eF_{\rm M}}{T_{\rm i}}(\phi-\bar{\phi})\right]dv^3=\delta n_{\rm e}, \tag{2}$$

where $\bar{\phi}$ is the gyrophase-averaged potential. It should be noted that the velocity space integral must be taken holding particle (not gyrocenter) position fixed. The perturbed electron density $\delta n_{\rm e}$ is assumed to be

$$\frac{\delta n_{\rm e}}{n_0}=\frac{e(\phi-\langle\phi\rangle)}{T_{\rm e}}, \tag{3}$$

where $\langle\cdots\rangle$ denotes the flux surface average.

2.2 Simulation domain and boundary conditions

In the δf framework, it is assumed that a steady equilibrium exists and satisfies the magneto-hydrodynamic equilibrium condition. Then, we can employ magnetic coordinates as configuration space coordinates, $\boldsymbol{r}=(x,y,z)$, and the equilibrium magnetic field is described as

$$\boldsymbol{B}=B_0\boldsymbol{\nabla}x\times\boldsymbol{\nabla}y=\frac{B_0}{\sqrt{g}}\frac{\partial\boldsymbol{r}}{\partial z}, \tag{4}$$

where $\sqrt{g}$ denotes the Jacobian. The flux-surface label x, the field-line label y and the field-aligned coordinate z correspond to the toroidal coordinates (r,θ,ζ) as $x=r-r_0$, $y=r_0(q\theta-\zeta)/q_0$, $z=\theta$ in a large-aspect-ratio tokamak with concentric circular magnetic flux surfaces, where q is the safety factor and the quantities with subscript 0 denotes the values at the center of the simulation domain. While plasma turbulence has short perpendicular wavelengths, its structure elongates in the direction parallel to the magnetic field. Therefore, a long and thin simulation domain along magnetic field lines is suitable for capturing the nature of plasma turbulence with reducing computational costs. An example of this flux-tube simulation domain is shown in Fig. 1, which is written by the projection of a box with short lengths in x and y and a long length in z. The flux-tube model is widely used to analyze turbulent transport in the local approximation limit, where the equilibrium quantities are given by local values.

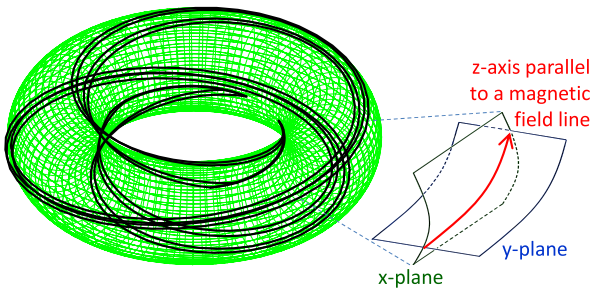

Fig. 1 An example of the flux-tube simulation domain and a schematic picture of the employed magnetic coordinates. A circular toroidal flux surface is also plotted as a reference.

By assuming statistically homogeneous turbulent fields, we impose periodic boundary conditions in x and y and apply the Fourier decomposition as

$$\phi(x,y,z) = \sum_{k_x}\sum_{k_y} \phi_{k_x,k_y}(z)\exp\left[i(k_x x + k_y y)\right]. \quad (5)$$

Additionally, there is the physical periodicity in the poloidal angle θ as $\phi(r,\theta,\zeta) = \phi(r,\theta+2\pi,\zeta)$. This leads the modified periodic boundary condition along the field-aligned coordinate z,

$$\phi_{k_x,k_y}(z) = \Theta\phi_{k_x+\Delta,k_y}(z+2\pi), \quad (6)$$

where the connection phase $\Theta = \exp(i2\pi k_y r_0)$ and connection wave number $\Delta = -2\pi s_0 k_y$ with the poloidal wave number k_y and the magnetic shear s_0 [14].

The equations (1)-(3) are numerically solved in $(k_x, k_y, z, v_\parallel, \mu)$ space except the nonlinear $\boldsymbol{E}\times\boldsymbol{B}$ advection term. Since direct calculations of nonlinear convolutions in wave number space are computationally too expensive, the $\boldsymbol{E}\times\boldsymbol{B}$ advection term is evaluated in the real space and transformed back to the wave number space by means of the 2D FFT and the 3/2 de-aliasing rule in (k_x, k_y).

3. Parallelization for Peta-Scale Computing

To attain good performance on a distributed-memory system, domain decomposition by using a MPI library is necessary, as well as thread parallelization. The original GKV code decomposes 5D phase space in three directions $(z, v_\parallel, \mu)$, which is not enough for multi-scale turbulence simulations. In order to achieve efficient computations on peta-scale supercomputers, we additionally decompose perpendicular wave-number space $\boldsymbol{k} = (k_x, k_y)$ and implement overlaps of computations and inter-node communications.

3.1 Domain decomposition

At the beginning, we briefly explain the parallelization in $(z, v_\parallel, \mu)$. The simulation domain is straightforwardly decomposed in 3D subdomains. Each subdomain has additional border cells to evaluate partial derivatives in $(z, v_\parallel, \mu)$ by using finite difference methods. The dominant communications are only six point-to-point communications between adjacent subdomains, which leads to excellent scaling. This 3D domain decomposition has already been implemented in the original GKV code. Extension to the 4D decomposition is done by performing the following wave-number-space decomposition on each subdomain in $(z, v_\parallel, \mu)$.

Now, we consider the 2D FFT in k_x and k_y, which are parallelized in k_y and k_x (x), respectively. Then, the $\boldsymbol{E}\times\boldsymbol{B}$ advection term is evaluated by using the 1D FFT in the following way:

- IF-X: 1D inverse FFT in k_x with k_y-decomposition.
- TR-XY: Data transpose from k_y-decomposition to x-decomposition.
- IF-Y: 1D inverse FFT in k_y with x-decomposition.
- RSC: Calculation of the $\boldsymbol{E}\times\boldsymbol{B}$ term in real space (x, y).
- FF-Y: 1D FFT in y with x-decomposition.
- TR-YX: Data transpose from x-decomposition to k_y-decomposition.
- FF-X: 1D FFT in x with k_y-decomposition.

For convenience, the abbreviations will be used throughout this paper. The inter-node transpose communications are implemented by using MPI_ALLTOALL, and the other terms are also parallelized in $(k_y, z, v_\parallel, \mu)$. We note that above wave-number space decomposition is better than another decomposition [i.e., (k_x, k_y) space is decomposed in k_x and (x, y) space is decomposed in y], which introduces extra point-to-point communications in k_x to apply the modified periodic boundary condition, Eq. (6). Since the distance of the mode connection is proportional to k_y, the number of the extra communications increases as resolutions or the number of parallelization in perpendicular wave-number space increase. Thus, the extra communications may degrade scalability if we employ the latter decomposition.

We use a FFT library, FFTW3 [15], for computing FFTs and employ a parallel de-aliasing strategy similar to that shown in Ref. [7]. For applying the 3/2 de-aliasing rule, k_x-direction is expanded before the nonlinear term calculation, and k_y-direction is expanded just after the first data transpose. Similarly, truncation in k_y is carried out before the second transpose, while truncation in k_x is done after the nonlinear term calculation. Since expansion and truncation in k_y are embedded in the parallel de-aliased spectral calculations, the costs of expansion and truncation in k_x only appear explicitly in the cost analysis (see Fig. 3 in Sec. 4).

3.2 Overlaps of FFTs and transpose

When one deals with multi-scale turbulence which requires high resolutions in wave-number space, the inter-node transpose communications in wave-number space account for a large part of computational costs and degrade scalability. To overcome the degradation of scalability due to the inter-node communications, overlaps of computations and communications are the most promising way. We implement the overlap method by employing a communication thread with the help of MPI/OpenMP hybrid parallelization. Figure 2 shows schematic pictures of the overlap method, where the master thread (the zeroth one) works as a communication thread and the others work as computation threads. In the case without overlaps, computation threads have to wait while the master thread performs inter-node communications, and after that, all threads carry out computations. In the case with overlap [see the right hand side of Fig. 2 (a)], the MASTER (or SINGLE) directive allows that communications on the master thread

Table 1 Overlap methods of FFTs and transpose. For simplicity, we use following abbreviations, IF-X (or -Y): Inverse FFT in x (or y), TR-XY (or -YX): Transpose communications from k_y (or k_x) -decomposition to k_x (or k_y) -decomposition, RSC: Calculations of the $\boldsymbol{E} \times \boldsymbol{B}$ advection term in real space, and FF-X (or -Y): Forward FFT in x (or y). Asterisks represent μ-loop and "(i)" denotes number of μ-loop.

No overlap	Partial overlaps (μ-loop splitting and overlaps of each transpose and neighboring FFTs.)	Integrated overlaps (μ-loop splitting and overlaps of all routines.)
* IF-X(i)	IF-X(1)	IF-X(1)
* TR-XY(i)	TR-XY(1), IF-X(2)	TR-XY(1), IF-X(2)
* IF-Y(i)	* TR-XY(i), IF-X($i+1$), IF-Y($i-1$)	TR-XY(2), IF-X(3), IF-Y(1)
* RSC(i)	TR-XY(n), IF-Y($n-1$)	* TR-XY(i), IF-X($i+1$), IF-Y($i-1$), RSC($i-2$)
* FF-Y(i)	IF-Y(n)	TR-XY(n), IF-Y($n-1$), RSC($n-2$), FF-Y(1)
* TR-YX(i)	* RSC(i)	TR-YX(1), FF-Y(2), IF-Y(n), RSC($n-1$)
* FF-X(i)	FF-Y(1)	TR-YX(2), FF-Y(3), FF-X(1), RSC(n)
	TR-YX(1), FF-Y(2)	* TR-YX(i), FF-Y($i+1$), FF-X($i-1$)
	* TR-YX(i), FF-Y($i+1$), FF-X($i-1$)	TR-YX(n), FF-X($n-1$)
	TR-YX(n), FF-X($n-1$)	FF-X(n)
	FF-X(n)	

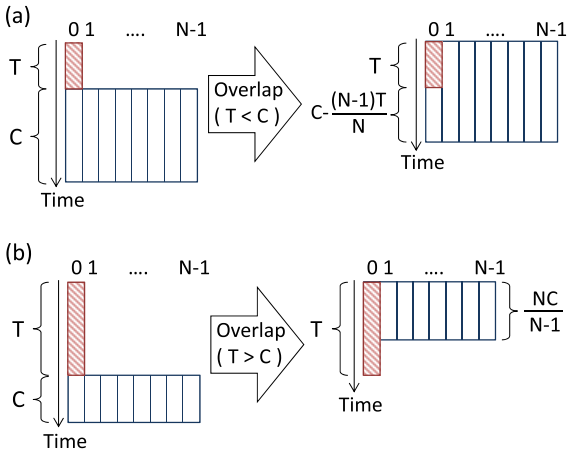

Fig. 2 Schematic pictures of the overlap method with MPI/OpenMP hybrid parallelization with N threads in the cases that (a) the communication cost T is smaller than the computational cost C, and (b) T is larger than C. The zeroth thread works as a communication thread, and the others work as computation threads. Shaded and unshaded boxes represent communication and computation tasks, respectively.

and computations, which are independent of the communication data, on the other threads are carried out at the same time. To reduce load imbalance, the computations are parallelized by means of DYNAMIC loop decompositions, where the chunk size is set to be 1 to decrease the granularity. The master thread can also carry out computations after communications end, if the communication costs are smaller than the computational costs. On the other hand, if communication costs are larger than computational ones, computation threads finish their works and wait until the communications end. Thus, the total cost S with the overlap of computations and communications is,

$$S = \begin{cases} C + \dfrac{T}{N} & \left(T \leq \dfrac{NC}{N-1}\right) \\ T & \left(T > \dfrac{NC}{N-1}\right) \end{cases}, \tag{7}$$

where N, T and C are the number of OpenMP threads, communication and computational costs without overlaps, respectively. We note that this is an ideal estimation, because computation tasks may not be uniformly parallelized by both of N and $N-1$ threads, which leads load imbalance and increases computational costs. While both cases in Fig. 2 show reductions of computational costs compared to the cost in the case without overlap, $C+T$, the case of $T < C$ is preferable from the view point of efficient use of computational resources. Therefore, it is important to find as many computing sections which are independent of the communication data as possible to improve scalability.

Let us consider the application of the above overlap method to FFTs and transpose. Fortunately, they are independent of μ (and z, $v_\parallel$), the decomposition of μ-loop makes overlaps of FFTs and transpose possible. Table 1 shows the newly developed overlap methods of FFTs and transpose. There are three levels of the parallelization strategies: the left column corresponds to the case without overlaps, which is same as the list shown in Sec. 3.1, the middle column represents the partial overlaps of each transpose and FFTs with neighboring indexes, $i \pm 1$, and the right column shows the integrated overlaps, where the computations between transpose communications are also overlapped. Although the partial overlaps mask communication costs only by $(n-1)/n$ parts of IF-X (or -Y) and FF-X (or -Y) (where n is the total number of μ-loop), the integrated overlaps mask communication costs by $(n-1)/n$

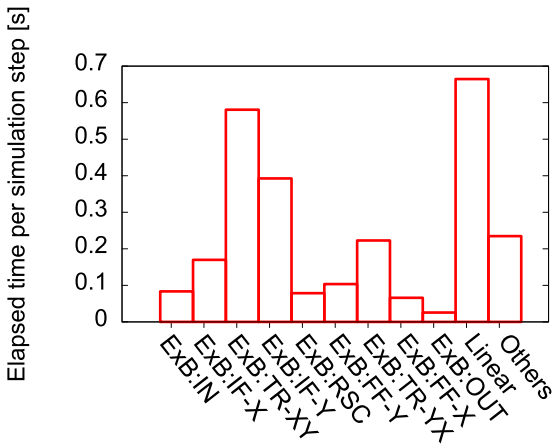

Fig. 3 Histogram of the elapsed time in the case without overlaps (where the number of the wave-number-space parallelization is 32). The items named "$\boldsymbol{E}\times\boldsymbol{B}$", "Linear", and "Others" correspond to the computational costs of the $\boldsymbol{E}\times\boldsymbol{B}$ advection term, linear terms and communications in $(z, v_\parallel, \mu)$, and the rest including a field solver. Abbreviations are same as Table 1 except IN: Expansion in k_x before data input to a FFT library, and OUT: Truncation in k_x of the output data from a FFT library.

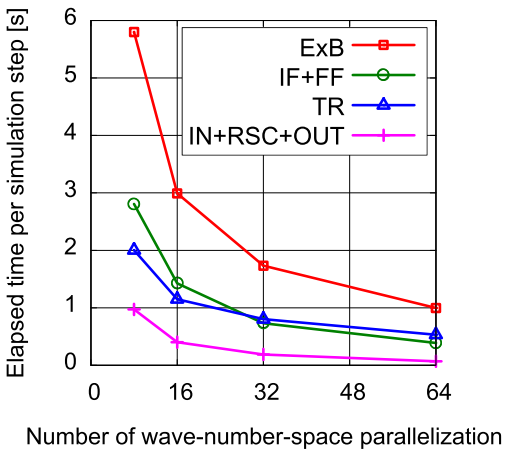

Fig. 4 Elapsed time of the $\boldsymbol{E}\times\boldsymbol{B}$ advection term calculation in the case without overlaps as a function of the number of wave-number-space parallelization. Elapsed time of total $\boldsymbol{E}\times\boldsymbol{B}$ term calculation, Fourier transform, transpose communications and the others are plotted with square, circular, triangle and cross dots, respectively.

parts of IF-X and FF-X, and the whole computation between the two transpose communications. Since the latter increases overlapped computations, the integrated overlaps are more promising to improve scalability than the partial overlaps.

4. Performance Analysis

We analyze effects of the presented overlap methods on the strong scaling of the calculation of the $\boldsymbol{E}\times\boldsymbol{B}$ advection term. Computations shown in this section were carried out on the FX10 supercomputer (SPARC64 IXfx 1.848 GHz, 14.78 GFlops/core, Memory bandwidth 5.3 GB/s/core, 16 cores/node, 6D Mesh/Torus interconnect, Interconnect bandwidth 5 GB/s × 4, 4800 nodes) in the University of Tokyo. In the following results, we employ $1024\times1024\times16\times16\times16$ grid points, which are scaled down from those required for multi-scale turbulence simulations, and decompose into $8(16, 32, 64)\times2\times2\times2$ subdomains. Each subdomain has a MPI process and eight OpenMP threads.

4.1 Analysis of computational costs

Before comparing the overlap methods, it is useful to check the computational costs of the target calculation without overlaps. Figure 3 shows the histogram of the computational cost of the GKV code employing the wave-number-space decomposition without overlaps of computations and communications. The calculation of the $\boldsymbol{E}\times\boldsymbol{B}$ advection term accounts for 66% of the total computational cost, and the costs of transpose communications and FFTs are dominant. In Fig. 4, elapsed time of the $\boldsymbol{E}\times\boldsymbol{B}$ advection term calculation is plotted as a function of the number of the wave-number-space parallelization. It is clearly shown that the cost of transpose communications decreases more slowly than that of computations. The cost of communications tends to be dominant as the number of wave-number space parallelization increases, and degrades scalability. In the case shown here, the cost of communications exceeds that of FFTs when the number of wave-number space parallelization is larger than 32, where the partial overlaps are not enough to mask communications.

Using Eq. (7), one can estimate the elapsed time in the case with the integrated overlaps from the data without overlaps. Since IF-X(0) and FF-X(n) are not overlapped, the estimation is given by

$$S_{\rm int} = \begin{cases} \dfrac{C_{\rm IF-X}+C_{\rm FF-X}}{n} + C_{\rm eff} + \dfrac{T}{N} & \left(T \le \dfrac{NC_{\rm eff}}{N-1}\right) \\ \dfrac{C_{\rm IF-X}+C_{\rm FF-X}}{n} + T & \left(T > \dfrac{NC_{\rm eff}}{N-1}\right) \end{cases}, \tag{8}$$

where $C_{\rm eff} = (n-1)(C_{\rm IF-X}+C_{\rm FF-X})/n + C_{\rm IF-Y} + C_{\rm RSC} + C_{\rm FF-Y}$ represents the effectively overlapped computational cost, and to simplify the problem, we only consider two cases: all communications are completely masked or not masked. When the data input and output for a FFT library are also included in the overlaps, the estimation becomes

$$S_{\rm int} = \max\left(C + \frac{T}{N}, \frac{C_{\rm IN}+C_{\rm IF-X}+C_{\rm FF-X}+C_{\rm OUT}}{n} + T\right). \tag{9}$$

This indicates that elapsed time can be reduced as the number of OpenMP threads N and of the pipelined loops n increase, if the overheads of communications are negligible.

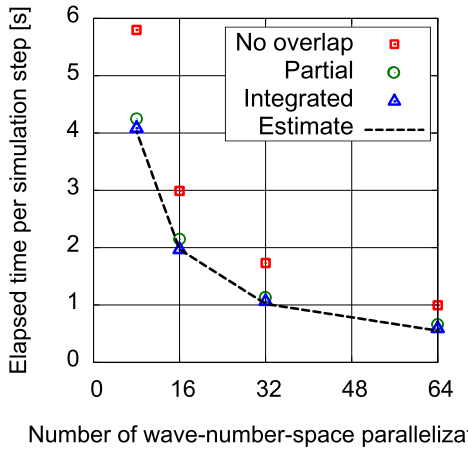

Fig. 5 Elapsed time of the $\boldsymbol{E} \times \boldsymbol{B}$ advection term calculation as a function of the number of wave-number-space parallelization. Square, circular and triangle dots correspond to the cases without overlaps, with the partial overlaps, and with the integrated overlaps, respectively. The estimation of the elapsed time in the case with the integrated overlaps, Eq. (9), is plotted as a dashed line.

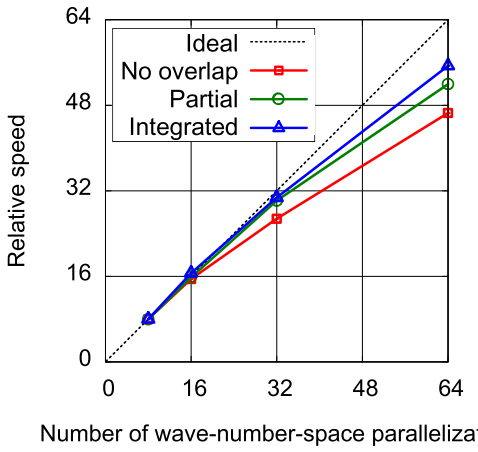

Fig. 6 Strong scaling of the $\boldsymbol{E} \times \boldsymbol{B}$ advection term calculation. The relative speeds (which are proportional to inverse of the elapsed time) in the cases without overlaps, with the partial overlaps, and with the integrated overlaps are plotted with square, circular, triangle dots, respectively. The dotted line represents the ideal speed up for reference.

4.2 Effects of the overlap methods

Elapsed time of the $\boldsymbol{E} \times \boldsymbol{B}$ advection term calculation in the cases without overlaps, with partial overlaps, and with integrated overlaps is plotted as a function of the number of wave-number-space parallelization in Fig. 5. The estimation of the elapsed time in the case with the integrated overlaps shows good agreements with the measured values, which assures that the integrated overlaps are successfully implemented with the smallest load imbalance. For example, the detailed time measurement for the case without overlaps records the computational cost $C = 1.827\,\mathrm{s}$, the communication cost $T = 1.149\,\mathrm{s}$ and $C_{\mathrm{eff}} = (n-1)(C_{\mathrm{IN}}+C_{\mathrm{IF-X}}+C_{\mathrm{FF-X}}+C_{\mathrm{OUT}})/n+C_{\mathrm{IF-Y}}+C_{\mathrm{RSC}}+C_{\mathrm{FF-Y}} = 1.738\,\mathrm{s}$, when the number of the wave-number-space parallelization is 16, the number of OpenMP threads $N = 8$ and the number of the pipelined loops $n = 8$. Since $T < NC_{\mathrm{eff}}/(N-1)$, it is expected that the integrated overlaps efficiently mask the communication cost. The elapsed time for the integrated overlaps is 1.968 s, which is the same as the estimated value ($S_{\mathrm{int}} = C + T/N = 1.971\,\mathrm{s}$) with a negligible error. It is demonstrated that the partial overlaps substantially reduce elapsed time from the case without overlaps, and the integrated overlaps achieve further speed-up. Figure 6 plots the inverse of the elapsed time normalized in each case. While the speed-up in the case without overlaps begins not to scale when the number of the wave-number-space parallelization is 32, where the communication cost becomes almost same as the computational cost, the cases with partial and integrated overlaps achieve 94% and 96% of ideal speed-up, respectively. The cases with overlaps also begin not to scale when the number of the wave-number-space parallelization is 64, because the communication costs are not completely masked [e.g., TR-XY(i) is masked by IF-X($i+1$), IF-Y($i-1$) and RSC($i-2$), but IF-X(1) cannot mask TR-XY(0)]. The results show that overlaps of computations and communications improve the strong scaling of parallel FFTs. Table 2 shows the performance of the $\boldsymbol{E} \times \boldsymbol{B}$ advection term calculation when the number of the wave-number-space parallelization is 32. Compared to the case without overlaps, the partial and integrated overlaps enhance computational performance by 56% and 63%, respectively. The advantage of the integrated overlaps clearly appears in Fig. 7, which plots speed-up compared to the case without overlaps. Speed-up due to the partial overlaps saturates when the number of the wave-number-space parallelization is 32. On the other hand, the integrated overlaps show further speed-up when the number of the wave-number-space parallelization is 64. This is because the integrated overlaps can mask a larger part of the communication cost than the partial overlaps. Therefore, we conclude that the integrated overlaps are more effective to improve the scalability of the $\boldsymbol{E} \times \boldsymbol{B}$ advection term calculation than the partial overlaps.

Table 2 Comparison of the performance of the $\boldsymbol{E} \times \boldsymbol{B}$ advection term calculation between the cases without overlaps, with the partial overlaps, and with the integrated overlaps (where the number of the wave-number-space parallelization is 32).

	TFlops	Peak ratio
No overlap	1.21	4.01%
Partial overlaps	1.86	6.17%
Integrated overlaps	1.97	6.54%

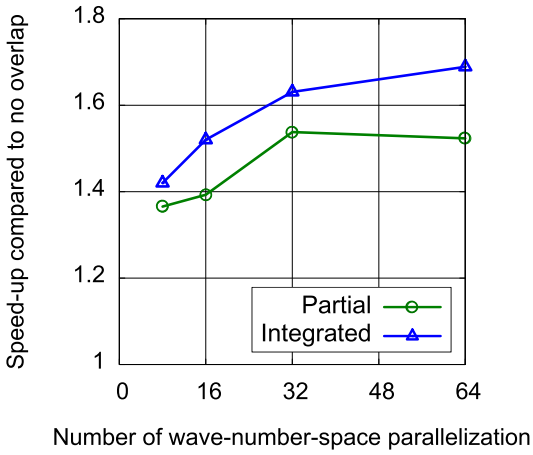

Fig. 7 Speed-up compared to the case without overlaps as a function of the number of wave-number-space parallelization. The circular and triangle dots correspond to the cases with partial and integrated overlaps, respectively.

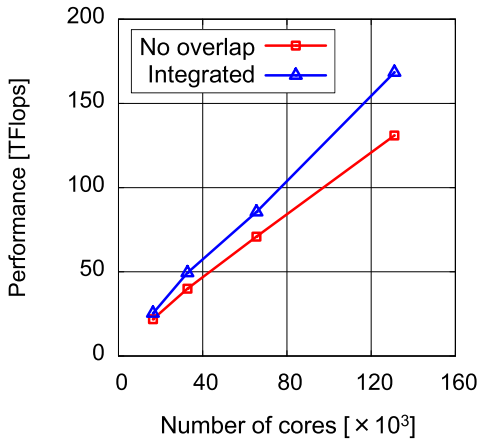

Fig. 8 Strong scaling test of the GKV code on the K computer [where $1024\times1024\times64\times64\times32$ grid is decomposed into $8(16, 32, 64)\times8\times8\times4$ subdomains and each subdomain employs 8 threads]. Square and triangle dots correspond to the cases without overlaps and with the integrated overlaps on the $\boldsymbol{E}\times\boldsymbol{B}$ term calculation, respectively.

5. Strong Scaling beyond 100 k Cores

We examine the impact of the developed overlap method on the total performance of the GKV code. The following computations are carried out on the K computer (SPARC64 VIIIfx 2 GHz, 16 GFlops/core, Memory bandwidth 8 GB/s/core, 8 cores/node, 6D Mesh/Torus interconnect, Interconnect bandwidth 5 GB/s × 4, 88128 nodes), employing the resolution required for the multi-scale turbulence simulations.

Figure 8 shows the strong scaling of the GKV code on the K computer. The domain decomposition in perpendicular wave number space allows us to employ more than 100 k cores, and the integrated overlaps significantly improve the strong scaling. In addition, the process mapping on a 3D torus network is optimized so that the transpose communications are performed among the neighboring nodes located in a 3D box shape, which maximize a bi-section bandwidth available on the torus network, and reduces costs of MPI_ALLTOALL. Thanks to the above optimization techniques, the GKV code achieves almost linear speed-up beyond 100 k cores. The parallel efficiency estimated from the Amdahl's law is 99.9998%, which is improved by an order of magnitude compared with the previous results (~ 99.998% evaluated from Fig. 2 shown in Ref. [16]). The effect of the overlaps becomes more significant as the number of cores increases. The case with the integrated overlaps achieves 168 TFlops (8.03% of the theoretical peak performance) at 131,072 cores, which is 29% higher performance than that in the case without overlaps. To further speed up the GKV code, additional tuning for serial and parallel algorithms will be performed in near future.

6. Conclusion

We have presented a massively-parallelized gyrokinetic Vlasov simulation code GKV, which is developed to study turbulent transport in magnetically-confined plasmas. To address multi-scale turbulence simulations, the parallelization of the GKV code has been extended to make it run efficiently on peta-scale supercomputers in two steps. First, three-dimensional domain decomposition has been extended to four-dimensional one to increase the number of available processes. This extension introduces additional inter-node transpose communications to calculate the nonlinear $\boldsymbol{E}\times\boldsymbol{B}$ advection term, which degrades scalability. Second, overlaps of computations and communications have been implemented to improve scalability by means of MPI/OpenMP hybrid parallelization. The integrated overlaps of whole calculations of the $\boldsymbol{E}\times\boldsymbol{B}$ advection term show better strong scaling than that in the case with partial overlaps of each transpose and neighbor FFTs, and achieves speed-up by 63% of that in the case without overlaps. The completed two extensions significantly improve scalability of the $\boldsymbol{E}\times\boldsymbol{B}$ advection term and make substantial speed-up. The strong scaling test on the K computer demonstrates that the extended GKV code is easily scaled up beyond 100 k cores and able to realize efficient computations of multi-scale turbulence simulations.

The newly developed overlap methods of FFTs and transpose communications are applicable if one finds independent multi-dimensional FFTs. For example, when one solves two-dimensional fluid equations including some unknowns (e.g., density, velocity, temperature, and so on) by a spectral method, FFTs of an unknown and transpose of another unknown can be easily overlapped.

Acknowledgments

This work is performed with supports of the HPCI Strategic Program for Innovative Research and the JAEA-NIFS Collaboration Program. A part of the results is obtained by early access to the K computer at the RIKEN Advanced Institute for Computational Science. One of the authors (S.M.) would like to thank H. Inoue and S. Tsutsumi for their supports to implement the parallel FFT algorithm.

[1] X. Garbet, Y. Idomura, L. Villard and T.-H. Watanabe, Nucl. Fusion **50**, 043002 (2010).
[2] T.-H. Watanabe and H. Sugama, Nucl. Fusion **46**, 24 (2006).
[3] D.G. Fox and S.A. Orszag, J. Comput. Phys. **11**, 612 (1973).
[4] J.W. Cooley and J.W. Tukey, Math. Comput. **19**, 297 (1965).
[5] C. Calvin, Parallel Comput. **22**, 1255 (1996).
[6] Z. Yin, L. Yuan and T. Tang, J. Comput. Phys. **210**, 325 (2005).
[7] M. Iovieno, C. Cavazzoni and D. Tordella, Comput. Phys. Commun. **141**, 365 (2001).
[8] P. Wapperom, A.N. Beris and M.A. Straka, Parallel Comput. **32**, 1 (2006).
[9] A. Danalis, K.Y. Kim, L. Pollock and M. Swany, Proc. IEEE/ACM Int. Conf. High Perform. Comput. (SC2005), Seattle, USA, pp. 58 (2005).
[10] P.D. Mininni, D. Rosenberg, R. Reddy and A. Pouquet, Parallel Comput. **37**, 316 (2011).
[11] T. Adachi, N. Shida, K. Miura, S. Sumimoto, A. Uno, M. Kurokawa, F. Shoji and M. Yokokawa, Comput. Sci. Res. Dev. **28**, 147 (2013).
[12] Y. Idomura, M. Nakata, S. Yamada, M. Machida, T. Imamura, T.-H. Watanabe, M. Nunami, H. Inoue, S. Tsutsumi, I. Miyoshi and N. Shida, Int. J. High Perform. Comput. Appl., DOI: 10.1177/1094342013490973 (2013).
[13] S. Maeyama, A. Ishizawa, T.-H. Watanabe, N. Nakajima, S. Tsuji-Iio and H. Tsutsui, Comput. Phys. Commun., DOI: 10.1016/j.cpc.2013.06.014 (2013).
[14] M.A. Beer, S.C. Cowley and G.W. Hammett, Phys. Plasmas **2**, 2687 (1995).
[15] M. Frigo and S.G. Johnson, Proc. IEEE **93**, 216 (2005).
[16] T.-H. Watanabe, Y. Todo and W. Horton, Plasma Fusion Res. **3**, 061 (2008).